Understanding the Role of Adaptivity in Machine Teaching: The Case of Version Space Learners

Yuxin Chen[†] **Adish Singla**[‡] **Oisin Mac Aodha**[†]
Pietro Porena[†] **Yisong Yue**[†]

[†]Caltech, {chenyux, macaodha, perona, yyue}@caltech.edu,
[‡]MPI-SWS, adishs@mpi-sws.org

Abstract

In real-world applications of education, an effective teacher adaptively chooses the next example to teach based on the learner's current state. However, most existing work in *algorithmic machine teaching* focuses on the batch setting, where adaptivity plays no role. In this paper, we study the case of teaching consistent, version space learners in an interactive setting. At any time step, the teacher provides an example, the learner performs an update, and the teacher observes the learner's new state. We highlight that adaptivity does not speed up the teaching process when considering existing models of version space learners, such as "worst-case" (the learner picks the next hypothesis randomly from the version space) and "preference-based" (the learner picks hypothesis according to some global preference). Inspired by human teaching, we propose a new model where the learner picks hypotheses according to some local preference defined by the current hypothesis. We show that our model exhibits several desirable properties, e.g., adaptivity plays a key role, and the learner's transitions over hypotheses are smooth/interpretable. We develop efficient teaching algorithms, and demonstrate our results via simulation and user studies.

1 Introduction

Algorithmic machine teaching studies the interaction between a teacher and a student/learner where the teacher's objective is to find an optimal training sequence to steer the learner towards a desired goal [31]. Recently, there has been a surge of interest in machine teaching as several different communities have found connections to this problem setting: (i) machine teaching provides a rigorous formalism for a number of real-world applications including personalized educational systems [30], adversarial attacks [19], and program synthesis [14]; (ii) the complexity of teaching ("Teaching-dimension") has strong connections with the information complexity of learning ("VC-dimension") [6]; and (iii) the optimal teaching sequence has properties captured by newly introduced models of interaction in machine learning, such as curriculum [4] and self-paced learning [20].

In the above-mentioned applications, adaptivity clearly plays an important role. For instance, in automated tutoring, adaptivity enables personalization of the content based on the student's current knowledge [25, 27]. In this paper, we explore the *adaptivity gain* in algorithmic machine teaching, i.e., how much speedup a teacher can achieve via adaptively selecting the next example based on the learner's current state? While this question has been well-studied in the context of active learning and sequential decision making [12], the role of adaptivity is much less understood in algorithmic machine teaching. A deeper understanding would, in turn, enable us to develop better teaching algorithms and more realistic learner models to exploit the adaptivity gain.

We consider the well-studied case of teaching a consistent, version space learner. A learner in this model class maintains a version space (i.e., a subset of hypotheses that are consistent with the examples received from a teacher) and outputs a hypothesis from this version space. Here, a hypothesis can be viewed as a function that assigns a label to any unlabeled example. Existing work

has studied this class of learner model to establish theoretical connections between the information complexity of teaching vs. learning [10, 32, 8]. Our main objective is to understand, when and by how much, a teacher can benefit by adapting the next example based on the learner's current hypothesis. We compare two types of teachers: (i) an *adaptive teacher* that observes the learner's hypothesis at every time step, and (ii) a *non-adaptive teacher* that only knows the initial hypothesis of the learner and does not receive any feedback during teaching. The non-adaptive teacher operates in a batch setting where the complete sequence of examples can be constructed before teaching begins.

Inspired by real-world teaching scenarios and as a generalization of the global "preference-based" model [8], we propose a new model where the learner's choice of next hypothesis $h' \in H'$ depends on some *local* preferences defined by the current hypothesis h. For instance, the local preference could encode that the learner prefers to make smooth transitions by picking a consistent hypothesis h' which is "close" to h. Local preferences, as seen in Fig. 1, are an important aspect of many machine learning algorithms (e.g., incremental or online learning algorithms [21, 22]) in order to increase robustness and reliability. We present results in the context of two different hypotheses classes, and show through simulation and user studies via efficient teaching algorithms that adaptivity can play a crucial role when teaching learners with local preferences.

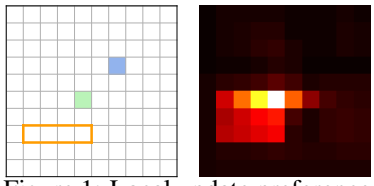


Figure 1: Local update preference. Users were asked to update the position of the orange rectangle so that green squares were inside and blue ones outside. The heatmap on the right displays the updated positions.

2 Related Work

Models of version space learners Within the model class of version space learners, there are different variants of learner models depending upon their anticipated behavior, and these models lead to different notions of teaching complexity. For instance, the (i) "worst-case" model [10] essentially assumes nothing and the learner's behavior is completely unpredictable, (ii) the "co-operative" model [32] assumes a smart learner who anticipates that she is being taught, and (iii) the "preference-based" model [8] assumes that she has a global preference over the hypotheses. Recently, some teaching complexity results have been extended to non-version space learners, such as Bayesian learners [29], probabilistic/randomized learners [24, 3], learners implementing an optimization algorithm [17], and for iterative learning algorithms based on gradient updates [18]. Here, we focus on the case of version space learners, leaving the extension to non-version space learners for future work.

Batch vs. sequential teaching Most existing work on algorithmic machine teaching has focused on the batch setting, where the teacher constructs a set of examples and provides it to the learner at the beginning of teaching [10, 32, 8, 5]. There has been some work on sequential teaching models that are more suitable for understanding the role of adaptivity. Recently, [18] studied the problem of iteratively teaching a gradient learner by providing a sequence of carefully constructed examples. However, since the learner's update rule is completely deterministic, a non-adaptive teacher with knowledge of the learner's initial hypothesis h^0 would behave exactly same as an adaptive teacher (i.e., the adaptivity gain is zero). [3] studied randomized version-space learners with limited memory, and demonstrated the power of adaptivity for a specific class of hypotheses. Sequential teaching has also been studied in the context of crowdsourcing applications by [15, 23], empirically demonstrating the improved performance of adaptive vs. non-adaptive teachers. However these approaches do not provide any theoretical understanding of the adaptivity gain as done in our work.

Incremental learning and teaching Our learner model with local preferences is quite natural in real-world applications. A large class of iterative machine learning algorithms are based on the idea of incremental updates which in turn is important for the robustness and generalization of learning [21, 22]. From the perspective of a human learner, the notion of incremental learning aligns well with the concept of the "Zone of Proximal Development (ZPD)" in the educational research and psychology literature [26]. The ZPD suggests that teaching is most effective when focusing on a task *slightly* beyond the current abilities of the student as the human learning process is inherently incremental. Based on these ideas of incremental learning, [2] studied the case of teaching a variant of a version space learner when restricted to incremental learning and is closest to our model with local preferences. However, there are two key differences in their model compared to ours: (i) they allow learners to select inconsistent hypotheses (i.e., outside the version space), (ii) the restricted movement in their model is a hard constraint which in turns means that teaching is not always feasible – given a problem instance it is NP-Hard to decide if a given target hypothesis is teachable or not.

3 The Teaching Model

We now describe the teaching domain, present a generic model of the learner and the teacher, and then state the teacher's objective.

3.1 The Teaching Domain

Let $\mathcal{X}$ denote a ground set of unlabeled examples, and the set $\mathcal{Y}$ denotes the possible labels that could be assigned to elements of $\mathcal{X}$. We denote by $\mathcal{H}$ a finite class of hypotheses, each element $h \in \mathcal{H}$ is a function $h : \mathcal{X} \to \mathcal{Y}$. In this paper, we will only consider boolean functions and hence $\mathcal{Y} = \{0, 1\}$. In our model, $\mathcal{X}$, $\mathcal{H}$, and $\mathcal{Y}$ are known to both the teacher and the learner. There is a *target hypothesis* $h^* \in \mathcal{H}$ that is known to the teacher, but not the learner. Let $\mathcal{Z} \subseteq \mathcal{X} \times \mathcal{Y}$ be the ground set of labeled examples. Each element $z = (x_z, y_z) \in \mathcal{Z}$ represents a labeled example where the label is given by the target hypothesis h^*, i.e., $y_z = h^*(x_z)$. Here, we define the notion of *version space* needed to formalize our model of the learner. Given a set of labeled examples $Z \subseteq \mathcal{Z}$, the version space induced by Z is the subset of hypotheses $\mathcal{H}(Z) \in \mathcal{H}$ that are consistent with labels of all the examples, i.e., $\mathcal{H}(Z) := \{h : h \in \mathcal{H} \texttt{ and } \forall z = (x_z, y_z) \in Z, h(x_z) = y_z\}$.

3.2 Model of the Learner

We now introduce a generic model of the learner by formalizing our assumptions about how she adapts her hypothesis based on the labeled examples she receives from the teacher. A key ingredient of this model is the *preference function* of the learner over the hypotheses as described below. As we show in the next section, by providing specific instances of this preference function, our generic model reduces to existing models of version space learners, such as the "worst-case" model [10] and the global "preference-based" model [8].

Intuitively, the preference function encodes the learner's transition preferences. Consider that the learner's current hypothesis is h, and there are two hypotheses h', h'' that they could possibly pick as the next hypothesis. We want to encode whether learner has any preference in choosing h' or h''. Formally, we define the preference function as $\sigma : \mathcal{H} \times \mathcal{H} \to \mathbb{R}_+$. Given current hypothesis h and any two hypothesis h', h'', we say that h' is preferred to h'' from h, iff $\sigma(h'; h) < \sigma(h''; h)$. If $\sigma(h'; h) = \sigma(h''; h)$, then the learner could pick either one of these two.

The learner starts with an initial hypothesis $h^0 \in \mathcal{H}$ before receiving any labeled examples from the teacher. Then, the interaction between the teacher and the learner proceeds in discrete time steps. At any time step t, let us denote the labeled examples received by the learner up to (but not including) time step t via a set Z^t, the learner's version space as $\mathcal{H}^t = \mathcal{H}(Z^t)$, and the current hypothesis as h^t. At time step t, we model the learning dynamics as follows: (i) the learner receives a new labeled example z^t; and (ii) the learner updates the version space $\mathcal{H}^{t+1}$, and picks the next hypothesis based on the current hypothesis h^t, version space $\mathcal{H}^{t+1}$, and the preference function σ:

$$h^{t+1} \in \{h \in \mathcal{H}^{t+1} : \sigma(h; h^t) = \min_{h' \in \mathcal{H}^{t+1}} \sigma(h'; h^t)\}. \tag{3.1}$$

3.3 Model of the Teacher and the Objective

The teacher's goal is to steer the learner towards the target hypothesis h^* by providing a sequence of labeled examples. At time step t, the teacher selects a labeled example $z^t \in \mathcal{Z}$ and the learner transitions from the current h^t to the next hypothesis h^{t+1} as per the model described above. Teaching finishes here if the learner's updated hypothesis $h^{t+1} = h^*$. Our objective is to design teaching algorithms that can achieve this goal in a minimal number of time steps. We study the *worst-case* number of steps needed as is common when measuring information complexity of teaching [10, 32, 8].

We assume that the teacher knows the learner's initial hypothesis h^0 as well as the preference function $\sigma(\cdot; \cdot)$. In order to quantify the gain from adaptivity, we compare two types of teachers: (i) an *adaptive teacher* who observes the learner's hypothesis h^t before providing the next labeled example z^t at any time step t; and (ii) a *non-adaptive teacher* who only knows the initial hypothesis of the learner and does not receive any feedback from the learner during the teaching process. Given these two types of teachers, we want to measure the *adaptivity gain* by quantifying the difference in teaching complexity of the *optimal adaptive* teacher compared to the *optimal non-adaptive* teacher.

4 The Role of Adaptivity

In this section, we study different variants of the learner's preference function, and formally state the adaptivity gain with two concrete problem instances.

4.1 State-independent Preferences

We first consider a class of preference models where the learner's preference about the next hypothesis does not depend on her current hypothesis. The simplest state-independent preference is captured by the "worst-case" model [10], where the learner's preference over all hypotheses are uniform, i.e., $\forall h, h', \sigma(h'; h) = c$, where c is some constant.

A more generic state-independent preference model is captured by non-uniform, global preferences. More concretely, for any $h \in \mathcal{H}$, we have $\sigma(h; h') = c_h \ \forall h' \in \mathcal{H}$, a constant dependent only on h. This is similar to the notion of the global "preference-based" version space learner introduced by [8].

Proposition 1 *For the state-independent preference, adaptivity plays no role, i.e., the sample complexities of the optimal adaptive teacher and the non-adaptive teacher are the same.*

In fact, for the uniform preference model, the teaching complexity of the adaptive teacher is the same as the *teaching dimension* of the hypothesis class with respect to teaching h^*, given by

$$\mathsf{TD}(h^*, \mathcal{H}) := \min_{Z} |Z|, \text{ s.t. } \mathcal{H}(Z) = \{h^*\}. \tag{4.1}$$

For the global preference model, similar to the notion of *preference-based teaching dimension* [8], the teaching complexity of the adaptive teacher is given by

$$\min_{Z} |Z|, \text{ s.t. } \forall h \in \mathcal{H}(Z) \backslash \{h^*\}, \sigma(h; \cdot) > \sigma(h^*; \cdot). \tag{4.2}$$

4.2 State-dependent Preferences

In real-world teaching scenarios, human learners incrementally builds up their knowledge of the world, and their preference of the next hypothesis naturally depends on their current state. To better understand the behavior of an adaptive teacher under a state-dependent preference model, we investigate the following two concrete examples:

Example 1 (2-REC) *$\mathcal{H}$ consists of up to two disjoint rectangles on a discretized grid and $\mathcal{X}$ represents nodes in the grid (cf. Fig. 1). Consider an example $z = (x_z, y_z) \in \mathcal{Z}$: $y_z = 1$ (positive) if the node x_z lies inside the target hypothesis, and 0 (negative) elsewhere.*

2-REC class is inspired by teaching a union of disjoint objects. Here, objects correspond to rectangles and any $h \in \mathcal{H}$ represents one or two rectangles. Furthermore, each hypothesis h is associated with a complexity measure given by the number of objects in the hypothesis. [7] recently studied the problem of teaching a union of disjoint geometric objects, and [1] studied the problem of teaching a union of monomials. Their results show that, in general, teaching a target hypothesis of lower complexity from higher complexity hypotheses is the most challenging task. This class has a natural notion of local preference, where a learner prefers to transition to a hypothesis with the same complexity as the current one. Furthermore, as elicited in Fig. 1, learners prefer smooth edits when learning a 2-REC hypothesis, e.g., by moving the smallest number of edges possible when changing their hypothesis.

Example 2 (LATTICE) *$\mathcal{H}$ and $\mathcal{X}$ both correspond to nodes in a d-dimensional integer lattice of length n. For a node v in the grid, we have an associated $h_v \in \mathcal{H}$ and $x_v \in \mathcal{X}$. Consider an example $z_v = (x_{z_v}, y_{z_v}) \in \mathcal{Z}$: $y_{z_v} = 0$ (negative) if the target hypothesis corresponds to the same node v, and 1 (positive) elsewhere. We consider the problem of teaching with positive-examples only.*

LATTICE class is inspired by teaching in a physical world from positive-only (or negative-only) reinforcements, for instance, teaching a robot to navigate to a target state by signaling that the current location is not the target. The problem of learning and teaching with positive-only examples is an important question with applications to learning languages, reinforcement learning, e.g, [9, 16]. For the LATTICE class, we assume that learner prefers to move to close-by hypothesis.

Theorem 2 *The ratio between the worst-case cost of the optimal non-adaptive teacher and the optimal adaptive teacher for teaching a 2-REC hypothesis is $\Omega\left(|h^0|/\log|h^0|\right)$, where $|h^0|$ denotes the number of positive instances induced by the learner's initial hypothesis h^0; the corresponding difference for teaching a LATTICE hypothesis is $\Omega(nd)$.*

In the above theorem we show that for both problems, under natural behavior of an incremental learner, adaptivity plays a key role. The proof of the theorem is built upon Lemma 5 and Lemma 6 in the next section; the detailed teaching algorithms achieving the bounds, is provided in the Appendix. Here, we highlight two necessary conditions under which adaptivity can possibly help: (i) preferences are local and (ii) there are ties among the learner's preference over hypotheses. The learner's current hypothesis, combined with the local preference structure, gives the teacher a handle to steer the learner in a controlled way.

5 Adaptive Teaching Algorithms

In this section, we first characterize the optimal teaching algorithm, and then propose efficient non-myopic algorithms for adaptive teaching.

5.1 The Optimality Condition

Assume that the learner's current hypothesis is h, and the current version space is $H \subseteq \mathcal{H}$. Let $D^*(h, H)$ denote the minimal number of examples required in the worst-case to teach h^*. We identify the following optimality condition for an adaptive teacher:

Proposition 3 *A teacher achieves the minimal teaching cost, if and only if for all learner's state (h, H), it picks an example such that*

$$z^* \in \arg\min_z \left(1 + \max_{h' \in \mathbf{C}(h,H,\sigma,z)} D^*\left(h', H \cap \mathcal{H}(\{z\})\right)\right)$$

where $\mathbf{C}(h, H, \sigma, z)$ denotes the set of candidate hypotheses in the next round as defined in (3.1)*, and for all (h, H), it holds that*

$$D^*(h, H) = \min_z \left(1 + \max_{h' \in \mathbf{C}(h,H,\sigma,z)} D^*\left(h', H \cap \mathcal{H}(\{z\})\right)\right)$$

In general, computing the optimal cost D^* for non-trivial preference functions, including uniform/global preference, requires solving a linear equation system of size $|\mathcal{H}| \cdot 2^{|\mathcal{H}|}$.

State-independent preference When the learner's preference is uniform, $D^*_u(h, H) = \mathsf{TD}(h^*, H)$ (Eq. 4.1) denotes the set cover number of the version space, which is NP-Hard to compute. A myopic approximate heuristic which gives best approximation guarantees for a polynomial time algorithm (logarithmic factor of the optimal cost [10]) is given by $\tilde{D}_u(h, H) = |H|$. For global preference, the optimal cost $D^*_g(h, H)$ is given by Eq. (4.2). i.e., the set cover number of all hypotheses in the version space that are more preferred over h^*. Similarly, one can also follow the greedy heuristic, i.e., $\tilde{D}_g(h, H) = |\{h' \in H : \sigma(h'; \cdot) \leqslant \sigma(h^*; \cdot)\}|$ to achieve a logarithmic factor approximation.

General preference Inspired by the two myopic heuristics above, we propose the following heuristic for general preference models:

$$\tilde{D}(h, H) = |\{h' \in H : \sigma(h'; h) \leqslant \sigma(h^*; h)\}| \tag{5.1}$$

In words, $\tilde{D}$ denotes the index of the target hypothesis h^* in the preference vector associated with h in the version space H. Notice that for the uniform (resp. global) preference model, the function $\tilde{D}$ reduces to $\tilde{D}_u$ (resp. $\tilde{D}_g$). In the following theorem, we provide a sufficient condition for the Algorithm 1 to attain provable guarantees:

Theorem 4 *Let h^* be the target hypothesis, and let $\bar{H}(\{z\}) = \{h \in H : h(x_z) \neq y_z\}$ be the set of hypotheses in H which are inconsistent with teaching example z. If for all (h, H) s.t. $h \in H$, the preference and the structure of the teaching examples satisfy:*

1. *$\forall h_i, h_j \in H$, $\sigma(h_i; h) \leqslant \sigma(h_j; h) \leqslant \sigma(h^*; h) \implies \sigma(h_j; h_i) \leqslant \sigma(h^*; h_i)$*
2. *$\forall H' \subseteq \bar{H}(\{z\})$, there exists $z' \in Z$, s.t., $\bar{H}(\{z'\}) = H'$.*

Then, the cost of Algorithm 1 which greedily minimizes (5.1) *is within a factor of $2(\log |\tilde{D}(h^0, \mathcal{H})| + 1)$ approximation of the cost of the optimal adaptive algorithm.*

We defer the proof of the theorem to the Appendix. Note that both the uniform preference model and global preference model satisfy Condition 1. Intuitively, the first condition states that there does not exist any hypothesis between h and h^* that provides a "short-cut" to the target. Condition 2 implies that we can always find teaching examples that ensure smooth updates of the version space. One of the settings for which the second condition holds is synthetic / constructive teaching [30], where the teacher can synthesize teaching examples (usually subject to some constraints). For instance, a feasible setting that fits Condition 2 is where we assume that the teacher can synthesize an example to remove any subset of hypotheses of size at most k, where k is some constant.

Algorithm 1 (Myopic) adaptive teaching

input: $\mathcal{H}$, σ, initial h^0, target h^*,
Initialize $t \leftarrow 0$, $Z^0 \leftarrow \varnothing$, $\mathcal{H}^0 \leftarrow \mathcal{H}$
while $h^t \neq h^*$ **do**
$z^t \in {}^1 \arg\min_z \left(1 + \max_{h' \in \mathbf{C}(h^t, \mathcal{H}(Z^t), \sigma, z)} \tilde{D}(.)\right)$
Learner makes an update; $t \leftarrow t + 1$
end while

Algorithm 2 (Non-myopic) adaptive teaching

input: $\mathcal{H}$, σ, initial h^0, target h^*, oracle S.
Initialize $t \leftarrow 0$, $Z^0 \leftarrow \varnothing$, $\mathcal{H}^0 \leftarrow \mathcal{H}$
while $h^t \neq h^*$ **do**
$z^{t+1} \leftarrow \mathsf{Teacher}(h^t, \mathcal{H}^t, \mathsf{S}(h^t, \mathcal{H}^t, h^*))$
Learner makes an update; $t \leftarrow t + 1$
end while

5.2 Efficient Non-Myopic Teaching Algorithms

When the conditions provided in Theorem 4 do not hold, the greedy heuristic (5.1) could perform very poorly.An important observation from Theorem 4 is that, when $\tilde{D}(h, H)$ is small, i.e., h^* is close to the learner's current hypothesis in terms of preference ordering, we need less stringent constraints on the preference function. This motivates adaptively devising intermediate target hypotheses to ground the teaching task into multiple, separate sub-tasks. Such divide-and-conquer approaches have proven useful for many practical problems, e.g., constructing a hierarchical decomposition for reinforcement learning tasks [13]. In the context of machine teaching, we assume that there is an oracle, $\mathsf{S}(h, H, h^*)$ that maps the learner's state (h, H) and the target hypothesis h^* to an intermediate target set, where the set of intermediate target hypotheses defines the current sub-task. We outline the generic algorithmic framework in Algorithm 2. Here, $\mathsf{Teacher}$ aims to provide teaching examples that brings the learner closer to one of the intermediate targets. In the adaptive setting, the oracle iteratively updates the intermediate hypothesis set. Below, we discuss different instances of $\mathsf{Teacher}$ for hypothesis classes 2-REC and LATTICE (with full details in the Appendix).

2-REC Let us consider the challenging case where the target hypothesis h^* represents a single rectangle, and the learner's initial hypothesis h^0 has two rectangles. As a typical example, imagine one of the two rectangles of h^0 (denoted by r_1) is overlapping with h^*, and the other (denoted by r_2) is disjoint with h^*. To teach h^*, the first sub-task (as provided by the oracle) is to eliminate the rectangle r_2 by providing negative examples so that the learner's hypothesis represents a single rectangle r_1. Then, the second sub-task (as provided by the oracle) is to teach h^* from r_1.

Lemma 5 *The adaptive greedy teacher requires at most* $\Theta(\log |r_2|)$ *examples to teach* h^**, while any non-adaptive teacher requires* $\Theta(|r_2|)$ *in the worst case.*

LATTICE Let $\text{dist}(h, h')$ denote the shortest length of all valid paths between h and h' along the integer lattice. We consider a distance-based oracle that goes through the learner's preference list from h^t, and returns the first hypothesis that is closer (measured by dist) to the target h^* than from h^t. In the Appendix, we show that for each sub-task generated by S, the conditions of Theorem 4 are satisfied. We show that the teaching complexity of Algorithm 2 for teaching LATTICE is as follows.

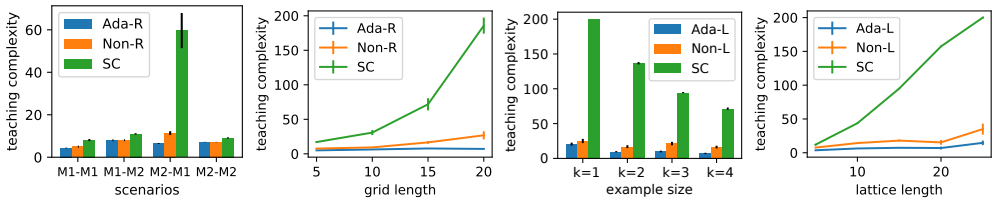
Lemma 6 *The adaptive teacher following the greedy heuristic* (5.1) *in Algorithm 2 requires* $d \cdot \textit{dist}(h^0, h^*)$ *examples, while any non-adaptive teacher requires at least* $(2d - 1) \cdot \textit{dist}(h^0, h^*)$.

6 Experiments

In this section, we empirically evaluate our teaching algorithms on the 2-REC and LATTICE tasks. We analyze our teaching algorithms via simulated learners, where we have full control over the teaching environment, so we can inspect the behavior of different algorithms and learner preference functions. Additional results showing robustness to noise are presented in the Appendix.

Experimental Setup The 2-REC hypothesis class consists of two subclasses, namely M_1: all hypotheses with one rectangle, and M_2: those with exactly two rectangles. We consider all four possible teaching scenarios, $M_{1\to1}$, $M_{1\to2}$, $M_{2\to1}$, $M_{2\to2}$, where each i, j in $M_{i\to j}$ specifies the subclasses of the learner's initial and the target hypothesis. In our simulations, we consider a fixed size discretized grid (varying from 5×5 to 30×30). The ground set of teaching examples consists

[1] Ties are broken by giving teaching examples that make the learner stay at the same hypothesis.



(a) 2-REC, size 15×15 (b) 2-REC, $M_{2\to1}$ (c) LATTICE, size 15×15 (d) LATTICE, $k = 3$

Figure 2: Simulation results for the 2-REC and LATTICE teaching tasks.

of all nodes on the grid. In each simulated teaching session, we sample a random pair of hypotheses (h^0, h^*) from the corresponding complexity classes as the learner's initial hypothesis and the target. For the LATTICE class, we consider finite 2-d integer lattices of fixed lengths. The (initial) version space consists of all nodes on the lattice. Rather than restricting each teaching example to remove only one hypothesis, in our experiments, we allow a teaching example to remove a connected region of up to size $k = 4$. In each teaching session, we set the target hypothesis h^* to be the center node of the integer lattice, and randomly sample a node on the lattice as the learner's initial hypothesis h^0.

Learner's preference The teaching and learning process follows the learner's dynamics as detailed in §3.2. To evaluate our teaching algorithm, we first simulate "ideal" learners for the two hypothesis classes. Upon seeing a teaching example, our simulated learner for the 2-REC class prefers to move the minimal number of edges possible to fit the data. To transition between different subclasses in 2-REC, we allow our simulated learner to (1) create a rectangle when the examples cannot be fit with a M_1 hypothesis, (2) eliminates a rectangle when all instances contained in the rectangle are inconsistent with the target hypothesis, and (3) merge two rectangles whenever the edges of two rectangles collapse on each other. For the LATTICE class, our simulated learner prefers to move to the closest valid hypothesis (i.e., the ones that are not covered by previous teaching examples).

Baselines We consider three types of teaching algorithms based on different models of the teacher and the learner. Each type is defined by $(\mathsf{L}, \mathsf{T}, \mathsf{I})$, where L is the preference model of the learner, $\mathsf{T} \in \{\sigma, \sigma_u\}$ is the preference function used by the teacher, and $\mathsf{I} \in \{\text{adaptive}, \text{non-adaptive}\}$ is the protocol of the teaching process. We simulated the following algorithms:

$(\mathsf{L} = \sigma, \mathsf{T} = \sigma, \text{adaptive})$ The learner and the teacher are consistent with the default state-dependent preference model σ, and the teacher picks adaptively. We denote the atomized teaching algorithm Teacher and oracle S in Algorithm 2 for both 2-REC and LATTICE as Ada-R and Ada-L.

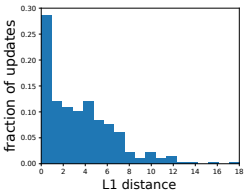
$(\mathsf{L} = \sigma, \mathsf{T} = \sigma, \text{non-adaptive})$ Both use the same σ, but the teacher's choice of examples only depends on the learner's initial state. We denote these non-adaptive algorithms as Non-R and Non-L for 2-REC and LATTICE, both matching the non-adaptive lower bounds provided in Theorem 2, with implementation details in the Appendix.

$(\mathsf{L} = \sigma, \mathsf{T} = \sigma_u, \text{non-adaptive})$ We assume the learner is consistent with σ, but the teacher acts according to the uniform preference model σ_u. The teaching process is non-adaptive by nature. Here, we run Algorithm 1, which reduces to the greedy set cover algorithm, denoted by SC.

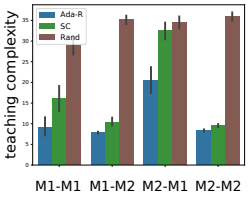
Results The performance of the algorithms are measured by their teaching complexity (the number of teaching examples required before the learner reaches h^*). For 2-REC, we run Ada-R, Non-R, and SC on all four teaching scenarios, over 50 trials with random samples of (h^0, h^*) . As we can see from Fig. 2a, Ada-R has a consistent advantage over the non-adaptive baselines across all four scenarios. As expected, teaching $M_{1\to1}$, $M_{1\to2}$, and $M_{2\to2}$ is easier, and the non-adaptive algorithms, such as SC perform well. In contrast, when teaching $M_{2\to1}$, we see a significant gain from Ada-R over the non-adaptive baselines. In the worst case, SC has to explore *all* the negative examples outside h^* to teach the target, whereas the non-adaptive teacher needs to provide all negative examples within the learner's initial hypothesis h^0 to make the learner jump from the subclass M_2 to M_1. In Fig. 2b, we observe that the adaptivity gain increases drastically as we increase the size of the grid. This matches our analysis of the logarithmic adaptivity gain in Theorem 2 for 2-REC. For LATTICE, we create different teaching scenarios by setting $k \in \{1, 2, 3, 4\}$ (which controls the maximal number of hypotheses an example can remove from the version space), and run Ada-R, Non-R, and SC over 50 random trials for each of these scenarios, see Fig. 2c. Although we only provide theoretical analysis for $k = 1$, we observe that Ada-L consistently has a big advantage over the non-adaptive baselines for all scenarios. In Fig. 2d, we fix $k = 3$ and increase the length of the lattice. Here, SC tries to cover the entire lattice, while Ada-L and Non-L focus on directing the learner to the optimal path (computed by our distances-based oracle). Similarly with the 2-REC case, the teaching complexity of Ada-L and Non-R matches our theoretical results.

h^*	Scenario	User Transition $M^u_{1\to1}$	$M^u_{1\to2}$	$M^u_{2\to1}$	$M^u_{2\to2}$
M_1	$M^t_{(1/2)\to(1/2)}$	0.54	0.13	0.06	0.27
M_2	$M^t_{(1/2)\to(1/2)}$	0.30	0.31	0.01	0.38
M_2	$M^t_{(1/2)\to2}$	0.00	0.61	0.00	0.39
M_2	$M^t_{2\to2}$	0.00	0.00	0.00	1.00
$M_{(1/2)}$	$M^t_{2\to(1/2)}$	0.00	0.00	0.11	0.89

(a) Transitions across classes



(b) Transitions $M^u_{1\to1}$



(c) Teaching results

Figure 3: (a) Participants prefer staying within the same hypothesis class when possible, displayed as the fraction of time they switched classes. $M^u_{1\to2}$ indicates that their hypothesis changed from one to two rectangles where M^t represents the valid possible configuration of hypotheses given the revealed squares, with $M_{(1/2)}$ indicating both are valid. (b) Participants favor staying at their current hypothesis if it remains valid, along with preferring smaller updates, computed as the $L1$ distance between the initial and updated rectangle. (c) Learners taught adaptively require less examples.

7 User Study

Here we describe experiments performed with real human participants from Mechanical Turk using the 2-REC task. We created an web-interface in order to (i) elicit the preference over hypotheses of real human learners, and to (ii) show the improved performance of adaptive teachers. Participants were asked to draw (or move) up to two rectangles on a grid of green, blue, or white squares. At each time step, the color of a white square selected by the teacher was revealed and the participants were asked to update their rectangle(s) (i.e., their current hypothesis) so only green squares were contained within them. The color of the revealed squares is defined by the target hypothesis h^*, see Fig. 1.

Real-world Human Preferences First we explore if participants exhibit any preference over possible transitions in the hypothesis space. We tested 215 participants, where each individual performed 10 trials on a grid of 12×12 squares. For each trial we randomly selected h^* from one of two classes M_1 or M_2 i.e., one or two target rectangles. Each participant was then presented with an initial set of revealed squares, a subset of all possible squares consistent with h^*, and asked to draw up to two valid rectangles. Once a valid hypothesis was selected by the participant, the interface updated the configuration of squares (either by adding or deleting squares) and participants were asked to update their hypothesis in light of the new information. We evaluated five different scenarios, corresponding to different combinations of initial and target hypotheses. In Fig. 3a we see that participants tend to favor staying in the same hypothesis class when allowed. Inside of the same hypothesis class, they have a preference towards updates that are close to their initial guess, see Fig. 3b.

Teaching Humans Next, we enlisted 200 participants to evaluate teaching algorithms on the 2-REC task. Each participant was first shown a blank 8×8 grid with either one or two initial rectangles, corresponding to h^0. There were randomly assigned to one of the following teaching algorithms: Rand (random selection), SC, or Ada-R. The target hypothesis, h^*, was selected to be either one or two rectangles, enabling us to explore four different combinations of initial and target hypotheses. This was repeated five times for each participant. For each trial, we recorded the number of examples required to learn the target hypothesis. Teaching was terminated when 60% of the squares were revealed. If the learner did not reach the target hypothesis by this time we set the number of teaching examples to this upper limit. An example teaching interaction is depicted in the Appendix. Fig. 3c illustrates the superiority of the adaptive teacher Ada-R, while random performs worst. In both cases where the target hypothesis is in M_2 the SC teacher performs nearly as well as the adaptive teacher, as at most 12 teaching examples are required to fully characterize the location of both rectangles. However, we observe a large gain from the adaptive teacher for $M_{2\to1}$.

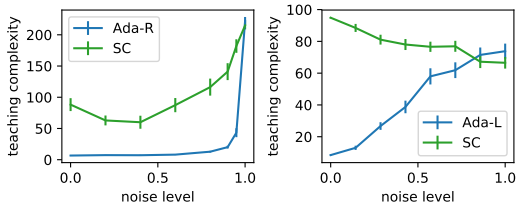
8 Conclusions

We explored the role of adaptivity in algorithmic machine teaching and showed that the adaptivity gain is zero when considering well-studied learner models (e.g., "worst-case" and "preference-based") for the case of version space learners. This is in stark contrast to real-life scenarios where adaptivity is an important ingredient for effective teaching. We highlighted the importance of local preferences (i.e., dependent on the current hypothesis) when the learner transitions to the next hypothesis. We presented hypotheses classes where such local preferences arise naturally, given that machines and humans have a tendency to learn incrementally. Furthermore, we characterized the structure of optimal adaptive teaching algorithms, designed near-optimal general purpose and application-specific adaptive algorithms, and validated these algorithms in simulation and with user studies.

Acknowledgments This work was supported in part by Northrop Grumman, Bloomberg, AWS Research Credits, Google as part of the Visipedia project, and a Swiss NSF Early Mobility Postdoctoral Fellowship.

A Supplemental Simulation Results

Here, we include additional simulation results, showing robustness to noise. In real-world teaching tasks, the learner's preference σ is not known to the teacher. In this experiment, we consider such noisy learners, whose preference σ' deviates from the preference σ of an "ideal" learner that the teacher is modeling. We simulate the noisy learners by randomly perturbing the preference σ of the "ideal" learner at each time step. With probability $1-\varepsilon$ the learner follows σ, and with probability ε, the learner switches to a random hypothesis in the version space. We consider the following algorithms: (1) ($\mathsf{L}=\sigma', \mathsf{T}=\sigma$, adaptive) i.e., Ada-R, Ada-L for the two hypothesis classes; and (2) ($\mathsf{L}=\sigma', \mathsf{T}=\sigma_u$, non-adaptive) i.e., SC. We evaluate the performance of these algorithms by varying the noise level ε in $[0,1]$. The results for 2-REC and LATTICE are shown in Fig. 4a and Fig. 4b. We observe that even for highly noisy learners e.g., $\varepsilon = 0.9$, Ada-R performs much better than the non-adaptive greedy strategy SC. This is because the gain of adaptivity is substantial at each time step for 2-REC. For the LATTICE class, where the adaptivity gain is less significant compared with 2-REC, our algorithm Ada-L becomes more sensitive to the noise in σ. Nevertheless, the adaptivity consistently outperforms SC in both cases (within standard error for stochastic noise).



(a) 2-REC, robustness (b) LATTICE, robustness

Figure 4: Robustness to noise. For both tasks we observe that the adaptive teacher is more robust across different noise levels compared to the non-adaptive SC approach.

B Supplemental Results from User Study

Example Teaching Traces Fig. 5 shows an example teaching session with the adaptive teacher, visualizing the teaching of two target rectangles from one initial rectangle i.e., $M_{1\to 2}$ At each time step, a new square is revealed by the teacher and the learner updates her hypothesis accordingly (depicted here as an orange rectangle).

Local Preference of Participants For eliciting the human update preferences depicted in Fig. 1 participants were shown a 10×10 grid and given an initial hypothesis and a subset of revealed squares. They were then instructed to update the position of the orange rectangle so that it contained green squares, with no blue squares inside it. They were free to draw up to two rectangles in total (including the initial hypothesis) and could move rectangles by clicking the center or grabbing the corners and dragging them to move the edges. They could also click to delete a rectangle and redraw it anywhere on the grid. The rectangles were fixed to only live on the grid lines. The task was completed when they submitted a valid configuration of the rectangles.

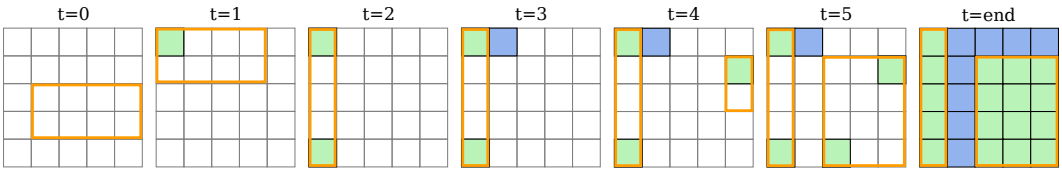
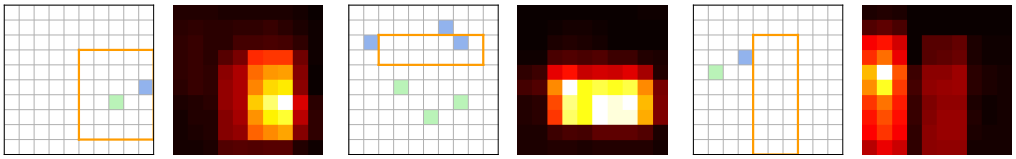


Figure 5: Adaptive teaching session. Here we see an example teaching session with the adaptive teacher Ada-R for a grid of size 5×5. The learner's hypothesis (in orange) is updated over time upon observing squares that are revealed by the teacher. In the final image we see the target hypothesis. For space reasons we omit some of the intermediate time steps.



(a) Example transition 1 (b) Example transition 2 (c) Example transition 3

Figure 6: Participants prefer smooth updates. For each pair, the left depicts the initial configuration of the grid shown to participants and the right is the heat map of their updated valid rectangles. Again, green squares had to lie within the rectange(s) and blue squares had to be outside.

100 participants on Mechanical Turk were each shown the same set of then initial configurations. We show the heatmap of their updated rectangles for three of the ten configurations in Fig. 6. Each position on the heatmap records the number of updated rectangles that overlap with that location, where brighter colors indicate more rectangles. We clearly see a preference for local updates as compared to the set of all possible, valid, updates.

C Proof of Theorem 4

In this section, we introduce useful notation and formally define the cost of a teaching algorithm. Then we prove the upper bound on the greedy cost as presented in Theorem 4.

Notations and formal definition of cost Let us use π to denote an adaptive teaching algorithm and ϕ to denote the internal randomness of the learner. We fix ϕ, the preference σ, the target hypothesis h^*, the learner's initial hypothesis h^0, and the version space $\mathcal{H}$. We denote the learner's hypothesis after running π for t steps as $h(\pi^t, \phi, \sigma, h^*, h^0, \mathcal{H})$. To be consistent with the main text of the paper, we use h^t as the shorthand notation for $h(\pi^t, \phi, \sigma, h^*, h^0, \mathcal{H})$ whenever it is unambiguous.
Given σ, h^*, h^0 and the version space $\mathcal{H}$, the worst-case cost of an algorithm π is formally defined as

$$\mathrm{cost}(\pi \mid \sigma, h^*, h^0, \mathcal{H}) = \max_{\phi} \min_{t} t, \text{ s.t., } h(\pi^t, \phi, \sigma, h^*, h^0, \mathcal{H}) = h^*.$$

We use $\mathrm{cost}^*(\sigma, h^*, h^0, \mathcal{H}) := \min_\pi \mathrm{cost}(\pi \mid \sigma, h^*, h^0, \mathcal{H})$ to denote the cost of an optimal algorithm, and use $\mathrm{cost}^{\mathrm{g}}(\sigma, h^*, h^0, \mathcal{H})$ to denote the cost of the greedy algorithm with respect to $\tilde{D}$ (Eq. (5.1)).

Preferred version space At time step t, we define the *preferred version space*

$$\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}^t) := \{h' \in \mathcal{H}^t : \sigma(h'; h^t) \leqslant \sigma(h^*; h^t)\} \tag{C.1}$$

to be the set of hypotheses that are more preferred over h^* (according to σ) from h^t.

Greedy vs. optimal We now proceed to the proof of Theorem 4. First, we provide a lower bound on the cost of the optimal algorithm.

Lemma 7 *Let σ_u be the uniform preference function. Assume that σ satisfies Condition 1 of Theorem 4. Then, the following inequality holds:*

$$\mathit{cost}^*(\sigma, h^*, h^0, \mathcal{H}) \geqslant \mathit{cost}^*(\sigma_u, h^*, h^0, \mathcal{S}(\sigma, h^*, h^0, \mathcal{H})).$$

Proof Fix ϕ. Denote the full teaching sequence of the optimal algorithm under preference σ as $Z^m = \{z^0, z^1, \ldots, z^{m-1}\}$, and the trace of the learner's hypotheses as $\{h^0, h^1, \ldots, h^{m-1}, h^*\}$. By definition, we have $\mathcal{H}(Z^t) = \mathcal{H}^t$.
Upon receiving teaching example z^t, the set of hypotheses eliminated from the preferred set at time step t is

$$\begin{aligned}
& \mathcal{S}(\sigma, h^*, h^t, \mathcal{H}^t) \backslash \mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^{t+1})) \\
= \; & \mathcal{S}(\sigma, h^*, h^t, \mathcal{H}^t) \backslash \{h' \in \mathcal{H}^{t+1} : \sigma(h'; h^t) \leqslant \sigma(h^*; h^t)\} \\
\overset{(a)}{=} \; & \mathcal{S}(\sigma, h^*, h^t, \mathcal{H}^t) \backslash \{h' \in \mathcal{H}^{t+1} : \sigma(h^{t+1}; h^t) \leqslant \sigma(h'; h^t) \leqslant \sigma(h^*; h^t)\} \\
\overset{(b)}{\supseteq} \; & \mathcal{S}(\sigma, h^*, h^t, \mathcal{H}^t) \backslash \{h' \in \mathcal{H}^{t+1} : \sigma(h'; h^{t+1}) \leqslant \sigma(h^*; h^{t+1})\} \\
= \; & \mathcal{S}(\sigma, h^*, h^t, \mathcal{H}^t) \backslash \mathcal{S}(\sigma, h^*, h^{t+1}, \mathcal{H}^{t+1})
\end{aligned} \tag{C.2}$$

Here, step (a) is due to the fact that $\mathcal{H}^{t+1} \cap \mathcal{S}(\sigma, h^{t+1}, h^t, \mathcal{H}^t) = h^{t+1}$, and step (b) follows from Condition 1 of Theorem 4.
Further observe that

$$\begin{aligned}\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}) &\subseteq \mathcal{S}(\sigma, h^*, h^0, \mathcal{H}) \backslash \mathcal{S}(\sigma, h^*, h^1, \mathcal{H}^1) \cup \mathcal{S}(\sigma, h^*, h^1, \mathcal{H}^1) \\ &\subseteq \bigcup_{t=0}^{m-1} \left(\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}^t) \backslash \mathcal{S}(\sigma, h^*, h^{t+1}, \mathcal{H}^{t+1})\right) \cup \mathcal{S}(\sigma, h^*, h^*, \mathcal{H}^m) \\ &= \bigcup_{t=0}^{m-1} \left(\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}^t) \backslash \mathcal{S}(\sigma, h^*, h^{t+1}, \mathcal{H}^{t+1})\right) \cup \{h^*\} \end{aligned} \tag{C.3}$$

Combining Eq. (C.2) and (C.3), we obtain

$$\begin{aligned}\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}) \backslash \{h^*\} &\subseteq \bigcup_{t=0}^{m-1} \left(\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}^t) \backslash \mathcal{S}(\sigma, h^*, h^{t+1}, \mathcal{H}^{t+1})\right) \\ &\subseteq \bigcup_{t=0}^{m-1} \left(\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}^t) \backslash \mathcal{S}(\sigma, h^*, h^t, \mathcal{H}^{t+1})\right)\end{aligned}$$

That is, providing teaching examples $\{z^1, z^2, \ldots, z^{m-1}\}$ is guaranteed to eliminate $\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}) \backslash \{h^*\}$. By definition, $\text{cost}^*(\sigma_u, h^*, h^0, \mathcal{S}(\sigma, h^*, h^0, \mathcal{H}))$ is the minimal number of examples required to eliminate $\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}) \backslash \{h^*\}$. Since the optimal cost is defined as the worst-case cost for all ϕ, it follows that $\text{cost}^*(\sigma_u, h^*, h^0, \mathcal{S}(\sigma, h^*, h^0, \mathcal{H})) \leqslant m \leqslant \text{cost}^*(\sigma, h^*, h^0, \mathcal{H})$. ■

In the following, we will focus on the analysis of the greedy algorithm with preference σ.

Lemma 8 *Assume that the preference function σ and the structure of tests satisfy Condition 1 and 2 from Theorem 4. Suppose we have run Algorithm 1 for m time steps. Let z^t be the current teaching example, $Z^t = \{z^0, \ldots, z^{t-1}\}$ be the set of examples chosen by the greedy teacher up to t, and h^t be the learner's current hypothesis. For any given example z, let h_z be the learner's next hypothesis assuming the teacher provides z. Then,*

$$\begin{aligned}&|\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t))| - |\mathcal{S}(\sigma, h^*, h^{t+1}, \mathcal{H}(Z^t \cup \{z^t\}))| \\ \geqslant\ &\frac{1}{2} \max_z \left(|\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t))| - |\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t \cup \{z\}))|\right)\end{aligned} \tag{C.4}$$

Proof Algorithm 1 picks the example which leads to the smallest preferred version space. That is,

$$z^t = \arg\min_z |\mathcal{S}(\sigma, h^*, h_z, \mathcal{H}(Z^t \cup \{z\}))|$$

If an example z is inconsistent with h^t, then by Condition 2 of Theorem 4, there exists an example z' which is consistent with h^t and only differs from z at h^t, i.e., $\mathcal{H}(\{z'\}) \backslash \mathcal{H}(\{z\}) = \{h^t\}$. By Condition 1 of Theorem 4 (and step (b) of Eq. (C.2)), we know $\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z\})) \subseteq \mathcal{S}(\sigma, h^*, h_z, \mathcal{H}(Z^t \cup \{z\}))$. Therefore

$$|\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z'\}))| - 1 = |\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z\}))| \leqslant |\mathcal{S}(\sigma, h^*, h_z, \mathcal{H}(Z^t \cup \{z\}))| \tag{C.5}$$

which gives us $|\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z'\}))| \leqslant \mathcal{S}(\sigma, h^*, h_z, \mathcal{H}(Z^t \cup \{z\}))| + 1$.
We first consider the following two cases.

C-1 $|\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z'\}))| = |\mathcal{S}(\sigma, h^*, h_z, \mathcal{H}(Z^t \cup \{z\}))| + 1$. Then, by Eq. (C.5), we have

$$\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z\})) = \mathcal{S}(\sigma, h^*, h_z, \mathcal{H}(Z^t \cup \{z\})).$$

That is, even the example z can bring the learner to a new hypothesis h_z, it does *not* introduce new hypotheses into the preferred version space.

C-2 $|\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z'\}))| < \mathcal{S}(\sigma, h^*, h_z, \mathcal{H}(Z^t \cup \{z\}))| + 1$. In this case, the greedy teacher will not pick z, because the gain of example z' is no less than the gain of z in terms of the greedy heuristic. In the special case where $|\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z'\}))| = \mathcal{S}(\sigma, h^*, h_z, \mathcal{H}(Z^t \cup \{z\}))|$, according to our tie-breaking rule in Algorithm 1, the teacher does not pick z, because it makes the learner move away from its current hypothesis and hence is less preferred.

For completeness, we also consider the case when the example z is consistent with h^t:

C-3 If the teacher picks the consistent example z, then the learner does not move away from her current hypothesis h^t. As a result, the preference ordering among set $\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z'\}))$ remains the same.

With the above three cases set up, we now reason about the gain of Algorithm 1. An important observation is that, the teaching examples provided by Algorithm 1 never add any hypotheses into the preferred version space. Therefore, at time step t, for any example z, we have

$$\begin{aligned}\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z\})) &= \mathcal{S}(\sigma, h^*, h^{t-1}, \mathcal{H}(Z^t \cup \{z\})) \\ &= \dots \\ &= \mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t \cup \{z\}))\end{aligned} \tag{C.6}$$

Next, we look into the gain for each of the three cases above.

C-1 Adding z^t changes the learner's hypothesis, i.e., $h^{t+1} \neq h^t$, but the resulting preferred version space induced by h^{t+1} is the same with that of h^t. In this case,

$$\begin{aligned}&|\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t))| - |\mathcal{S}(\sigma, h^*, h^{t+1}, \mathcal{H}(Z^t \cup \{z^t\}))| \\ =\ &|\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t))| - \min_z |\mathcal{S}(\sigma, h^*, h^{t+1}, \mathcal{H}(Z^t \cup \{z\}))| \\ =\ &|\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t))| - \min_z |\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t \cup \{z\}))| \\ =\ &\max_z \big(|\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t))| - |\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t \cup \{z\}))|\big)\end{aligned} \tag{C.7}$$

C-2 In this case, we have

$$\begin{aligned}|\mathcal{S}(\sigma, h^*, h^{t+1}, \mathcal{H}(Z^t \cup \{z^t\}))| &= |\mathcal{S}(\sigma, h^*, h_z, \mathcal{H}(Z^t \cup \{z\}))| \\ &= |\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z'\}))|\end{aligned}$$

and Algorithm 1 picks $z^t = z'$ according to the tie-breaking rule. The learner does not move away from her current hypothesis: $h^{t+1} = h^t$. However, since $\mathcal{H}(\{z'\}) \backslash \mathcal{H}(\{z\}) = \{h^t\}$, we get

$$\begin{aligned}|\mathcal{S}(\sigma, h^*, h^{t+1}, \mathcal{H}(Z^t \cup \{z^t\}))| &= |\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z'\}))| \\ &= |\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z\}))| + 1 \\ &\overset{(a)}{=} \min_{z''} |\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t \cup \{z''\}))| + 1 \\ &\overset{\text{(C.6)}}{=} \min_{z''} |\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t \cup \{z''\}))| + 1\end{aligned}$$

where step (a) is due to the greedy choice of Algorithm 1. Further note that before reaching h^*, the gain of Algorithm 1 is positive. Therefore,

$$\begin{aligned}&|\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t))| - |\mathcal{S}(\sigma, h^*, h^{t+1}, \mathcal{H}(Z^t \cup \{z^t\}))| \\ \geqslant\ &\frac{1}{2}\big(1 + |\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t))| - |\mathcal{S}(\sigma, h^*, h^{t+1}, \mathcal{H}(Z^t \cup \{z^t\}))|\big) \\ =\ &\frac{1}{2}\max_z \big(|\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t))| - |\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t \cup \{z\}))|\big)\end{aligned} \tag{C.8}$$

C-3 In this case, z^t is consistent with h^t, the greedy gain amounts to the maximal number of hypotheses removed from the preferred version space. Thus we have

$$\begin{aligned}&|\mathcal{S}(\sigma, h^*, h^t, \mathcal{H}(Z^t))| - |\mathcal{S}(\sigma, h^*, h^{t+1}, \mathcal{H}(Z^t \cup \{z^t\}))| \\ =\ &\max_z \big(|\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t))| - |\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t \cup \{z\}))|\big)\end{aligned} \tag{C.9}$$

Combining Eq. (C.7), (C.8), (C.9) finishes the proof. ■

Proof of Theorem 4 We are now ready to provide the proof for Theorem 4.

Proof [Proof of Theorem 4]

Based on the discussions in Lemma 8, we know that the teaching sequence provided by Algorithm 1 never adds new hypotheses into the initial preferred version space $\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t))$, and neither does it move *consistent* hypotheses out of $\mathcal{S}(\sigma, h^*, h^0, \mathcal{H}(Z^t))$. The teaching objective thus reduces to a set cover objective, and the teaching finishes once all hypotheses, except h^*, in the initial preferred version space are covered.

In Lemma 8, we show that at each time step, the gain of Algorithm 1 is at least $\frac{1}{2}$ the gain of the greedy set cover algorithm. Therefore, Algorithm 1 is a 2-approximate greedy set cover algorithm [28]. The logarithmic approximation result then follows from [28, 11]:

$$\text{cost}^{\text{g}}(\sigma, h^*, h, \mathcal{S}) \leqslant 2\left(\log|\tilde{D}(h^0, \mathcal{H})| + 1\right)\text{cost}^*(\sigma_u, h^*, h^0, \mathcal{S}(\sigma, h^*, h^0, \mathcal{H})). \tag{C.10}$$

Combining Eq. C.10 with Lemma 7 completes the proof. ■

D 2-REC

In this section, we provide the detailed specification of the 2-REC hypothesis class introduced in §4, and present the adaptive algorithm Ada-R and non-adaptive algorithm Non-R. We finish this section with the analysis of the two algorithms, which we use to prove Lemma 5.

D.1 Subclasses and Preference Structure

As described earlier, 2-REC contains two (non-overlapping) subclasses M_1 and M_2 with different complexity. Let $\mathcal{H}^t$ be the learner's version space at time step t, and h^t be the learner's current hypothesis. We use $M(h^t)$ to denote which subclass h^t is in.

D.1.1 "Shortcuts" between M_1 and M_2

We consider two special subsets of hypotheses in M_2.

M_1 spin-off: In the first special subset, each hypothesis can be considered as a *spin-off* from a M_1 hypothesis: Given $h \in M_2$, we call h a *spin-off* of $h' \in M_1$, if and only if one of the rectangles in h is fully aligned with h', and the other rectangle contains a singleton instance. Let $r_1 : M_2 \to M_1$ (resp. r_2) denote the function that maps a hypothesis $h \in M_2$ to the first (resp. second) rectangle it contains. Then the set of all M_1 spin-offs is

$$S_1 = \{h \in M_2 : h \text{ is a } M_1 \text{ spin-off }\} = \{h \in M_2 : |r_1(h)| = 1 \vee |r_2(h)| = 1\}. \tag{D.1}$$

M_1 splits: In the second special subset, each hypothesis can be considered as a *split* from a M_1 hypothesis: Given $h \in M_2$, we call h a M_1 *split*, if and only if there exists no other M_2 hypothesis in the minimal rectangle that encloses h. We denote the set of all M_1 split as S_2:

$$S_2 = \{h \in M_2 : h \text{ is a } M_1 \text{ split }\}. \tag{D.2}$$

We consider the subsets S_1 and S_2 as shortcuts between as M_1 and M_2. In the following, we will describe our preference model of the learners, based on the subclasses previously defined.

D.1.2 The Preference Structure

For any pair of hypotheses h, h' from the *same subclass*, define $\text{dist}_e(h, h')$ to be the minimal number of edge movements required to move from h to h'. For example, $\max_{h,h' \in M_1} \text{dist}_e(h, h') \leqslant 4$ and $\max_{h,h' \in M_2} \text{dist}_e(h, h') \leqslant 8$.

Our general assumption about the learner's preference structure σ for 2-REC is that learners prefer to make small moves among hypotheses. In Fig. 7, we depict the valid transitions among 2-REC hypotheses upon receiving an example. That includes the following cases:

C-1 $h^t \in M_1$. In this case, the learner prefers hypotheses in M_1, then hypotheses in S_1, and lastly hypotheses in $M_2 \backslash S_1$. More specifically,

(a) Within M_1, the learner prefers a hypothesis with smaller distance from h^t;

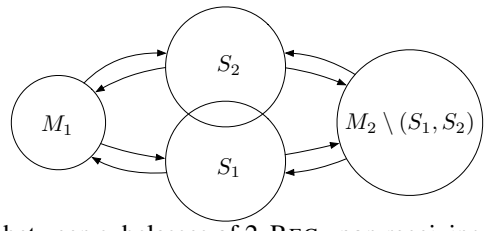


Figure 7: Transitions between subclasses of 2-REC upon receiving *one* teaching example

(b) Within S_1, the learner has uniform preferences
—if learner makes a jump to S_1, it corresponds to "draw new rectangle" operation.

(c) Within $M_2 \backslash S_1$, the learner also has uniform preferences from h^t.
Note that case (c) is not needed for designing an adaptive teacher, because the learner always needs to move to a hypothesis in $S_1 \cup S_2$ from h^t, prior to moving to $M_2 \backslash (S_1 \cup S_2)$.

C-2 $h^t \in S_1 \cup S_2$. In this case,

(a) The current hypothesis is always the most preferred.

(b) Next,

i. if $h^t \in S_1$, the learner prefers hypotheses in M_1 which share a rectangle with h^t —this corresponds to a "delete a rectangle" operation.

ii. if $h^t \in S_2$, the learner prefers the hypothesis in M_1 which is the minimal rectangle that encloses h^t
—this corresponds to a "merge two existing rectangles" operation.

(c) Hypotheses further down in the preference list are M_2 hypotheses. These hypotheses are ordered by their distances $\text{dist}_e(\cdot, h^t)$ towards h^t.

(d) All other hypotheses in M_1 are least preferred.

(e) Within M_1, the learner prefers the ones that overlap with one of its rectangles.

C-3 $h^t \in M_2 \backslash (S_1 \cup S_2)$. In this case, the learner prefers hypotheses in M_2 over M_1. More specifically,

(a) Within M_2, the learner prefers a hypothesis with smaller distance.

(b) Within M_1, the learner has uniform preferences.
Note that under such preference, the learner always needs to move to a hypothesis in S prior to moving to M_1.

Intuitively, our oracle generates the intermediate hypotheses from the "short-cut" hypothesis set.

D.2 Proof of Lemma 5

Here, we provide the proof of Lemma 5, by presenting the two teaching algorithms, namely Ada-R (which is instantiated from Algorithm 2 with the Ada-R-Teacher presented in Algorithm 3 designed for the 2-REC hypothesis class) and Non-R (a non-adaptive algorithm achieving the same order of the optimal non-adaptive cost).

D.2.1 Ada-R

The oracle An essential component in Algorithm 2 is the oracle $\mathsf{S}(h^t, \mathcal{H}^t, h^*)$, which defines the intermediate target hypotheses at each time step. For Ada-R, we employ the following adaptive oracle. Consider the four teaching scenarios:

$M_{1\to1}$: $h^* \in M_1 \wedge h^t \in M_1$, we have $\mathsf{S}(h^t, \mathcal{H}^t, h^*) = \{h^*\}$.

$M_{2\to2}$: $h^* \in M_2 \wedge h^t \in M_2$, we have $\mathsf{S}(h^t, \mathcal{H}^t, h^*) = \{h^*\}$.

$M_{1\to2}$: $h^* \in M_2 \wedge h^t \in M_1$, in this case,

1. If $h^t = r_1(h^*) \vee h^t = r_2(h^*)$, $\mathsf{S}(h^t, \mathcal{H}^t, h^*) = \{h \in S_1 : h^t = r_1(h) \vee h^t = r_2(h)\}$.
2. Otherwise, $\mathsf{S}(h^t, \mathcal{H}^t, h^*) = \{r_1(h^*)\}$, where $r_1(h^*)$ denotes the first rectangle of h^*.

$M_{2\to1}$: Now let us consider the case $h^* \in M_1 \wedge h^t \in M_2$.

1. If both rectangles in h^t overlap with h^*:

(a) If h^t is a split of h^*, $\mathsf{S}(h^t, \mathcal{H}^t, h^*) = \{h^*\}$.
(b) Otherwise, the oracle returns a subset of hypotheses of S_2, where (1) each hypothesis h is a split of h^*, and (2) each of the two rectangles contained in h overlaps with exactly one rectangle in h^t.
For discussion simplicity, let us refer to such subset as the set of valid splits of h^*.

2. If at least one of the rectangles in h^t is disjoint with h^*,
(a) If $h^t \in S_1$, $\mathsf{S}(h^t, \mathcal{H}^t, h^*) = \{h \in M_1 : h^t \text{ is a spin-off of } h\}$
(b) Otherwise, the oracle returns a subset of hypothesis of S_1, where each hypothesis contains (1) a rectangle that fully aligns with one of the rectangles that are disjoint with h^*, and (2) another singleton rectangle which is in the other rectangle of h^t.

The adaptive teacher A useful observation is that for teaching $M_{1\to1}, M_{2\to2}$, and $M_{1\to2}$, an optimal teacher needs to provide at most 12 teaching examples:

- To teach $M_{1\to1}$, it is sufficient to provide the two positive corner instances in the diagonal positions (say, the lower left corner and the upper right corner), and the two adjacent negative instances for each of the positive corners—this amounts to 6 examples in total.
- To teach $M_{2\to2}$ and $M_{1\to2}$, it is sufficient to provide 6 corner examples for each of the rectangles—this amounts to 12 examples in total.

There are two implications from the above observation. First, instances lie on the diagonal corners are useful for teaching targets from the same subclass. Second, even though one can design smart algorithms for teaching the above cases (via adaptivity and exhaustive search), we are not likely to benefit from it by much. Therefore, in such cases, Ada-R-Teacher goes through the, at most 12, candidate corner point candidates and proposes an example that brings the learner closer to the target hypothesis.

The more challenging, yet inspiring case, is $M_{2\to1}$. To bring the learner to the intermediate targets, Ada-R-Teacher runs a greedy heuristic derived from Eq. (5.1): it picks an example z so that after the learner makes a move, the number of hypotheses before reaching the closest h^* is the minimal:

$$z^* \in \arg\min_z \min_i |\{h' \in \mathcal{H}^t \cap \mathcal{H}(\{z\}) : \sigma(h'; h_z) \leqslant \sigma(h_i^*; h_z)\}|. \tag{D.3}$$

Here, h_z denotes the learner's next hypothesis if provided with teaching example z.

Now, let us go through each case to analyze the performance of the above greedy heuristic.

- When the learner's hypothesis is at Scenario $[M_{2\to1}]$–1–(a) or $M_{2\to1}$–2–(a), the learner is ready to make a jump to M_1. A *single* example suffices to achieve this, and hence the greedy heuristic is optimal.
- When the learner's hypothesis is at Scenario $[M_{2\to1}]$–1–(b), the goal of teaching is to reach any of the hypothesis in $\mathsf{S}(h^t, \mathcal{H}^t, h^*)$—the set of valid splits of h^*. Here, we consider two different cases:
  1. Either of the two rectangles of h^t is not aligned with h^* on exactly 3 edges.
  In this case, Ada-R-Teacher picks examples from the corner instances of h^* to bring the edges of two rectangles to h^*. In the worst case, we need all 12 corner instances of h^* to ensure that.
  2. Both rectangles of h^t are aligned with h^* on exactly 3 edges.
  In this case, the distance from h^t to any valid splits of h^* is 1. Ada-R-Teacher follows the greedy heuristic to pick the next example. Note that before reaching the target, the distance between any hypothesis of the learner its closest target remains to be 1. Therefore, the greedy heuristic (Eq. D.3) leads to a binary search algorithm. Let the maximal length of h^* be ℓ, then the teacher needs $O(\log|\ell| + 1)$ examples in the worst case to eliminates *all* the intermediate targets.
- When the learner's hypothesis is at Scenario $[M_{2\to1}]$–2–(b), the goal of teaching reduces to reaching any of the hypotheses in $\mathsf{S}(h^t, \mathcal{H}^t, h^*)$ by providing *negative* examples in the rectangle which contains the singleton intermediate targets. To be consistent with the notation in Lemma 5, we refer to such rectangle by r_2. It is not difficult to see that no matter what examples the teacher picks, the distances (defined by dist_e) from the resulting hypothesis of the learner to any of the intermediate target hypothesis are equal. Hence the learner's preference over the intermediate target hypotheses is uniform, and the greedy objective (Eq. D.3) leads to a binary search algorithm. Therefore, the teacher needs $O(\log|r_2| + 1)$ examples in the worst case to eliminates *all* the intermediate targets.

The pseudo code of Ada-R-Teacher is given in Algorithm 3.

Algorithm 3 Ada-R-Teacher: the adaptive teacher for 2-REC (subroutine for Ada-R/Algorithm 2)

input: $\mathcal{H}$, σ, current h^t, selected examples Z^t, targets $\mathsf{S}(h^t, \mathcal{H}, h^*) = \{h_1^*, \ldots, h_k^*\}$
if $(h^* \in M_1 \wedge h^t \in M_2)$ **then**
  if $\{h_1^*, \ldots, h_k^*\}$ are M_1 splits from h^* and $\min_i \text{dist}(h^0, h_i^*) > 1$ **then**
    $T \leftarrow$ GenerateAllCorners(h^*)
    $z \leftarrow$ Sample$(T \backslash Z^t)$
  else
    $z \leftarrow \arg\min_z \min_i |\{h' \in \mathcal{H}(z) : \sigma(h'; h_z) \leqslant \sigma(h_i^*; h_z)\}|$
  end if
else
  T $\leftarrow$ GenerateDiagonalCorners(h_1^*)
  $z \leftarrow$ Sample$(T \backslash Z^t)$
end if
output: next teaching example z

Algorithm 4 Non-R: the non-adaptive teaching algorithm for 2-REC

input: $\mathcal{H}$, σ, initial h^0, selected examples Z^t, oracle $\mathsf{S}(h^0, \mathcal{H}, h^*) = \{h_1^*, \ldots, h_k^*\}$
if $(h^* \in M_1 \wedge h^0 \in M_2)$ **then**
  if $\{h_1^*, \ldots, h_k^*\}$ are M_1 splits from h^* **then**
    $T_1 \leftarrow$ GenerateAllCorners(h^*)
    $T_2 \leftarrow$ GenerateAllEdgeInstances(h^*)
    {↳ *provide all the (positive) teaching examples on the edges/borders of h^* to make the learner "merge" the two rectangles in h^0.*}
    $Z \leftarrow (T_1, T_2)$
  else
    $T_1 \leftarrow$ GenerateAllConsistentInstances($r(h^0)$ which contains the singleton rectangles)
    {↳ *if one of the rectangles of h^0 is disjoint with h^*, provide all the examples inside this rectangle to make the learner "delete" it.*}
    $T_2 \leftarrow$ GenerateAllCorners(h^*)
    $Z \leftarrow (T_1, T_2)$
  end if
else
  $Z \leftarrow$ GenerateDiagonalCorners(h_1^*)
end if
output: Sequence of teaching examples Z

The non-adaptive teacher Next, we present the non-adaptive algorithm, Non-R, which is used in our simulation.

According to our modeling assumption, other than the learner's initial hypothesis, the non-adaptive teacher does not observe how the learner updates her hypotheses. However, this does not affect teaching the easy scenarios, namely $M_{1\to1}$, $M_{2\to2}$, and $M_{1\to2}$. In such cases, the non-adaptive teacher provides all the diagonal corner examples (including both positive and negative) as described earlier, which needs at most 6 examples for M_1 target, and 12 for M_2.

When teaching $M_{2\to1}$, in particular, for the case of $[M_{2\to1}]$–1–(b) and $[M_{2\to1}]$–2–(b), it is not possible for the non-adaptive teacher to perform a binary search. The reason is that the learner's behavior is highly non-deterministic at every iteration, and the uncertainty of the learner's hypotheses diffuses at an exponential rate. The best thing a non-adaptive teacher can do (in the worst case) is a linear scan over the candidate teaching examples, in which case it requires $\Omega(|\ell|)$ examples for $[M_{2\to1}]$–1–(b), and $\Omega(|r_2|)$ examples for $[M_{2\to1}]$–2–(b).

The pseudocode of Non-R is provided in Algorithm 4.

E LATTICE

In this section, we specify the LATTICE hypothesis class, and present the details of the adaptive algorithm Ada-L-Teacher (which is used to instantiate Ada-L from Algorithm 2) and the non-adaptive algorithm Non-R, and provide the proof for Lemma 6.

Algorithm 5 Non-L: the non-adaptive teaching algorithm for LATTICE

input: $\mathcal{H}$, σ, initial h^0
$P \leftarrow \text{NodesOnShortestPath}(h^0, h^*, \mathcal{H})$
$T \leftarrow \text{NeighboringNodes}(P)$
$Z \leftarrow (T \backslash P, P)$
output: Sequence of teaching examples Z

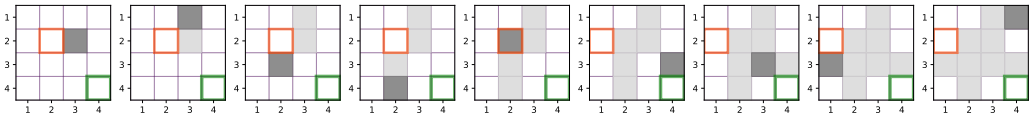
E.1 Preference Structure and The Greedy Teacher

Preference function Recall that hypotheses in LATTICE are represented by its nodes, and the preference function is defined through the L_1 distance: $\sigma(h'; h) = L_1(h', h)$. To facilitate understanding of the concept class, one can imagine a toy scenario, where the teacher aims to teach/steer a human learner to reach a goal state in a physical environment. Each hypothesis/node corresponds to some unexplored territory, and there exists an example which flags the territory as explored. The learner prefers local moves, and if all neighboring territories are explored, the learner jumps to the next closest one.

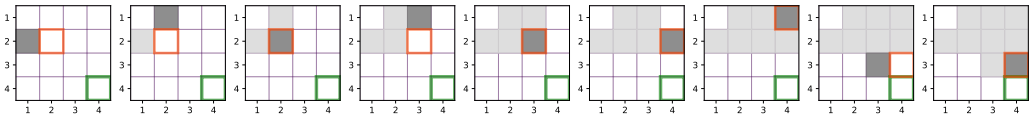
A failure case for Algorithm 1 We show in Fig. 8 that the myopic teacher Algorithm 1 can perform poorly on simple teaching tasks. As we can see in the first plot of Fig. 8a, the learner starts at square $h^0 = (2, 2)$ and the target hypothesis is at $h^* = (4, 4)$. The target hypothesis is the least preferred hypothesis from h^0. Any teaching example will not change the rank of h^*:

- Any example other than $(2, 2)$ will not move the learner away from h^0, hence the gain in $\tilde{D}(h^0, \mathcal{H})$ is 1.
- Meanwhile, the teaching example $(1, 1)$ moves the learner away from $(1, 1)$. However, when computing the greedy heuristic we are considering the gain in the worst-case—which corresponds to the case where the learner jumps to $(1, 2)$ or $(2, 1)$, and the gain is 1.

In this particular example, the gradient of the objective function (Eq.(5.1)) is 0, which provides no information as to which example to select next. Hence, the myopic teacher uses random instances to teach; in the worst case, the performance can be arbitrarily bad, even compared with a non-adaptive policy.



(a) Myopic - Algorithm 1



(b) Ada-L– Algorithm 2 (with the oracle defined by Eq. (E.1))

Figure 8: Teaching sequences generated by Algorithm 1 and Algorithm 2 (Ada-L) on a 4×4 lattice. The learner's initial hypothesis is marked by orange, and the target is marked by green. The dark gray square represents the teaching example at the current time step, while light gray squares represent the previous teaching examples.

E.2 Proof of Lemma 6

We now provide the details of Ada-L and Non-L, based on which we prove Lemma 6.

The oracle As we have briefly discussed in §5, we consider a distance-based oracle, which returns the first hypotheses that are closer (measured by the length of the shortest path) to the target h^* than from h^t. Formally, we define

$$\begin{aligned} \mathsf{S}(h^t, \mathcal{H}, h^*) = &\{h' \in \mathcal{H} : \forall h'' \in \mathcal{H}, \sigma(h''; h^t) \leqslant \sigma(h'; h^t) \\ &\implies \text{dist}(h'', h^*) = \text{dist}(h', h^*) \ \vee \\ &\quad \text{dist}(h', h^*) < \text{dist}(h^t, h^*) \leqslant \text{dist}(h'', h^*)\} \end{aligned} \tag{E.1}$$

The adaptive teacher Assume that the learner starts on a 2-d lattice, with all nodes unexplored. Then, at time step 0, since there is no other node blocking the way from h^0 to h^*, our oracle returns the adjacent nodes to h^0 on the lattice in the direction of h^*, as the intermediate target hypotheses. At time step t, if any of the adjacent nodes of h^0 in the opposite direction of h^* (henceforth are referred as "bad neighbor") is unexplored, the teacher will pick an example which explores/blocks it first, before exploring the current hypothesis itself. The reason is that,

- since the teacher progressively advances to the target hypothesis h^*, it will never pick (random) teaching examples that explore all the adjacent nodes of the bad neighbors of h^t, before that bad neighboring node is explored. See Fig. 8b as an example.
- If a bad neighbor of h^t has an unexplored neighbor other than h^t, then Ada-L will pick an example that explores the bad neighbor first before exploring the learner's current hypothesis (as going to the bad neighbor leads to lower gain in terms of the greedy objective, and hence we want to explore it before the learner jumps there).
- If h^t has no bad neighbor, then the teacher provides an example that explores h^t itself, and the learner proceeds to the next intermediate target node.

From the above discussion, we know that the total number of nodes needs to be explored before reaching h^* is $((2-1)+1)\cdot \text{dist}(h^0, h^*) = 2\text{dist}(h^0, h^*)$ (see Fig. 8b, last plot). The same reasoning generalizes to $d-$dimensional lattice as well, where an adaptive teacher needs to explore the bad neighbors in d directions, and progress in one direction at each round. Hence, the cost of Ada-L is of order $O\left(d \cdot \text{dist}(h^0, h^*)\right)$.

The non-adaptive teacher Our strategy for designing a non-adaptive neighbor follows closely from the adaptive strategy. Since the uncertainty of the teaching process comes from the learner randomly jumping to neighboring nodes of equal preference, we aim to pre-compute a set of teaching example that ensures the learner taking a deterministic path. Compared to the adaptive strategy which only explores "bad" neighbors, the non-adaptive teacher also has to explore "good"neighbors to form a deterministic path from h^0 to h^*. In $d-$dimension, there are $2d$ directions in total, with one of them containing a good path. Therefore, the teacher needs at least $(2d-1)\text{dist}(h^0, h^*)$, which finishes the proof of Lemma 6.

References

[1] Frank J Balbach. Measuring teachability using variants of the teaching dimension. *Theoretical Computer Science*, 397(1-3):94–113, 2008.

[2] Frank J Balbach and Thomas Zeugmann. Teaching learners with restricted mind changes. In *ALT*, pages 474–489. Springer, 2005.

[3] Frank J Balbach and Thomas Zeugmann. Teaching randomized learners with feedback. *Information and Computation*, 209(3):296–319, 2011.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pages 41–48, 2009.

[5] Yuxin Chen, Oisin Mac Aodha, Shihan Su, Pietro Perona, and Yisong Yue. Near-optimal machine teaching via explanatory teaching sets. In *AISTATS*, April 2018.

[6] Thorsten Doliwa, Gaojian Fan, Hans Ulrich Simon, and Sandra Zilles. Recursive teaching dimension, vc-dimension and sample compression. *JMLR*, 15(1):3107–3131, 2014.

[7] Ziyuan Gao, David Kirkpatrick, Christoph Ries, Hans Simon, and Sandra Zilles. Preference-based teaching of unions of geometric objects. In *ALT*, pages 185–207, 2017.

[8] Ziyuan Gao, Christoph Ries, Hans U Simon, and Sandra Zilles. Preference-based teaching. *JMLR*, 18(31):1–32, 2017.

[9] E Mark Gold. Language identification in the limit. *Information and control*, 10(5), 1967.

[10] Sally A Goldman and Michael J Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

[11] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *JAIR*, 42:427–486, 2011.

[12] Lisa Hellerstein, Devorah Kletenik, and Patrick Lin. Discrete stochastic submodular maximization: Adaptive vs. non-adaptive vs. offline. In *CIAC*, pages 235–248, 2015.

[13] Bernhard Hengst. *Hierarchical Reinforcement Learning*, pages 495–502. 2010.

[14] Susmit Jha and Sanjit A. Seshia. A theory of formal synthesis via inductive learning. *Acta Inf.*, 54(7):693–726, 2017.

[15] Edward Johns, Oisin Mac Aodha, and Gabriel J Brostow. Becoming the expert-interactive multi-class machine teaching. In *CVPR*, pages 2616–2624, 2015.

[16] Steffen Lange and Thomas Zeugmann. Incremental learning from positive data. *Journal of Computer and System Sciences*, 53(1):88–103, 1996.

[17] Ji Liu and Xiaojin Zhu. The teaching dimension of linear learners. *JMLR*, 17(162):1–25, 2016.

[18] Weiyang Liu, Bo Dai, Ahmad Humayun, Charlene Tay, Chen Yu, Linda B. Smith, James M. Rehg, and Le Song. Iterative machine teaching. In *ICML*, pages 2149–2158, 2017.

[19] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI*, pages 2871–2877, 2015.

[20] Deyu Meng, Qian Zhao, and Lu Jiang. A theoretical understanding of self-paced learning. *Inf. Sci.*, 414:319–328, 2017.

[21] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International journal of computer vision*, 77(1-3):125–141, 2008.

[22] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2), 2012.

[23] Adish Singla, Ilija Bogunovic, G Bartók, A Karbasi, and A Krause. On actively teaching the crowd to classify. In *NIPS Workshop on Data Driven Education*, 2013.

[24] Adish Singla, Ilija Bogunovic, Gábor Bartók, Amin Karbasi, and Andreas Krause. Near-optimally teaching the crowd to classify. In *ICML*, pages 154–162, 2014.

[25] Cem Tekin, Jonas Braun, and Mihaela van der Schaar. etutor: Online learning for personalized education. In *ICASSP*, pages 5545–5549, 2015.

[26] Lev Vygotsky. Zone of proximal development. *Mind in society: The development of higher psychological processes*, 5291:157, 1987.

[27] Daniel S Weld, Eytan Adar, Lydia Chilton, Raphael Hoffmann, Eric Horvitz, Mitchell Koch, James Landay, Christopher H Lin, and Mausam Mausam. Personalized online education—a crowdsourcing challenge. In *HCOMP*, 2012.

[28] Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[29] Xiaojin Zhu. Machine teaching for bayesian learners in the exponential family. In *Advances in Neural Information Processing Systems*, pages 1905–1913, 2013.

[30] Xiaojin Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*, pages 4083–4087, 2015.

[31] Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N. Rafferty. An overview of machine teaching. *CoRR*, abs/1801.05927, 2018.

[32] Sandra Zilles, Steffen Lange, Robert Holte, and Martin Zinkevich. Models of cooperative teaching and learning. *JMLR*, 12(Feb):349–384, 2011.